**Note at Github: Github_version**

R08943016 吳諺倫

# Week 1

## Video 1

### What is RStudio and Why Should You Download it?

> R Studio
>
> > Make the environment for R nesser
> >
> > Extend the R
> >
> > More console
> >
> > > Environment
> > >
> > > - Import Dataset
> > > - Variable Block - Working Memory
>
> > > Console
> > >
> > > - Files
> > > - Plots
> > > - Packages
> > > - Help
> > > - Viewer
> > >
> > > History

```
x<-1:5
y<-1:5
plot(x,y)
ls()
```

```
# Create a new variable
z <- 11:15
# Add up x,y,z
sum(x,y,z)
```

- File → New File → R Markdown
  - Allows you to embed R code and R output directly into documents.
- File → New Project
  - Allows you to manage all your files and output related to a project in one spot.

Reference:

What is RStudio and Why Should You Download It? | R Tutorial 1.1 | MarinStatsLectures

---

# Week 1

## Video 2

### Installing R and RStudio

> Install R
>
> > Free open source for Windows, macOS and linux-system
> >
> > R project website
> >
> > > Download
> > >
> > > - Download in one of CRAN pages
> > > - Select the nearest location
> > > - Taiwan
> > >
> > >   R website in NTU CSIE
> > > - Select your operating system
> > > - Download the installation file and select the default options
>
> Install RStudio
>
> > RStudio Website
> >
> > > Download
> > >
> > > - Click on **DOWNLOAD FREE DESKTOP IDE**
> > > - Select RStudio Desktop
> > > - The recommended one usually will be adequately

---

Reference:

Download and Install R and RStudio | R Tutorial 1.2 | MarinStatsLectures

---

# Week 1

## Video 3

### Getting Started with R, Part I

```
# Assign 11 to value x by equal sign =
> x = 11
# R is case sensitive
> X
Error: object 'X' not found
# Use arrow <- to assign value
```

```
> y <- 7
> y
[1] 7
# Overwritten y by value 9
> y <- 9
> y
[1] 9
# See the value in workspace or ls command to ask R the data stored in workspace
> ls()
[1] "x" "y"
# Remove an object using rm() command
> rm(y)
> y
Error: object 'y' not found
# Variable can include period and number
> x.1 <- 14
> x.1
[1] 14
# But number cannot be in the first character
> 1x <- 22
Error: unexpected symbol in "1x"
# Assign charcater values to variable
> xx <- "marin"
> xx
[1] "marin"
# Treat as character if we use quotations
> yy <- "1"
> yy
[1] "1"
> 11+14
[1] 25
> 7*9
[1] 63
> [x]
[1] 11
> x+y
[1] 20
> z <- x+y
> z
[1] 20
> y^2
[1] 81
> x^2 + y^2
[1] 202
> sqrt(y)
[1] 3
> y^(1/2)
[1] 3
> log(y)
[1] 2.197725
> exp(y)
[1] 8103.084
> log2(y)
[1] 3.169925
> abs(-14)
[1] 14
# Incomplete command
> sqrt(y
+
+ )
[1] 3
# Use arrow kep up will bring you to previous command
# Use # for command
```

---

Reference:

[Getting started with R: Basic Arithmetic and Coding in R | R Tutorial 1.3 | MarinStatsLectures]()

---

# Week 2

## Video 4

### Getting Started with R, Part II: Creating vectors, matrices, and performing some simple operations on them

```
# Create a vector with c or concatenate command
> x1 <- c(1,3,5,7,9)
> x1
[1] 1 3 5 7 9
# Create a vector of character elements by including "" around the elements
> gender <- c("male", "female")
> gender
[1] "male"  "female"
# Create a sequence of integer values using the colon (:)
> 2:7
[1] 2 3 4 5 6 7
# For more general case, use the "seq" command
> seq(from=1, to=7, by=1)
[1] 1 2 3 4 5 6 7
# A sequence running from 1 to 7 in increments of a third
> seq(from=1, to=7, by=1/3)
 [1] 1.000000 1.333333 1.666667 2.000000 2.333333 2.666667 3.000000
 [8] 3.333333 3.666667 4.000000 4.333333 4.666667 5.000000 5.333333
[15] 5.666667 6.000000 6.333333 6.666667 7.000000
# Use the "rep" command to create a vector of repeated numbers or characters
> rep(1, times=10)
[1] 1 1 1 1 1 1 1 1 1 1
# Repeate the characters "marin" and repeat 5 times
> rep("marin", times=5)
[1] "marin" "marin" "marin" "marin" "marin"
# Repeate a sequence multiple times
> rep(1:3, times=5)
[1] 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3
> rep(seq(from=2, to=5, by=0.25), times=5)
 [1] 2.00 2.25 2.50 2.75 3.00 3.25 3.50 3.75 4.00 4.25 4.50 4.75 5.00
[14] 2.00 2.25 2.50 2.75 3.00 3.25 3.50 3.75 4.00 4.25 4.50 4.75 5.00
[27] 2.00 2.25 2.50 2.75 3.00 3.25 3.50 3.75 4.00 4.25 4.50 4.75 5.00
[40] 2.00 2.25 2.50 2.75 3.00 3.25 3.50 3.75 4.00 4.25 4.50 4.75 5.00
[53] 2.00 2.25 2.50 2.75 3.00 3.25 3.50 3.75 4.00 4.25 4.50 4.75 5.00
> rep(c("m", "f"), times=5)
```

```
[1] "m" "f" "m" "f" "m" "f" "m" "f" "m" "f"
>
> x <- 1:5
> y <- c(1,3,5,7,9)
>
# Add an value to each element of the vector using the "plus" command
> x+10
[1] 11 12 13 14 15
# Minus an value to each element of the vector using the "minus" command
> x-10
[1] -9 -8 -7 -6 -5
# Multiply an value to each element of the vector using the "multiply" command
> x*10
[1] 10 20 30 40 50
# Divide an value to each element of the vector using the "divide" command
> x/2
[1] 0.5 1.0 1.5 2.0 2.5
# If two vectors are of the same length, we may add/substract/mult/div #
corresponding elements
> x
[1] 1 2 3 4 5
> y
[1] 1 3 5 7 9
> x+y
[1]  2  5  8 11 14
> x-y
[1]  0 -1 -2 -3 -4
> x*y
[1]  1  6 15 28 45
> x/y
[1] 1.0000000 0.6666667 0.6000000 0.5714286 0.5555556

# Extract elements of a vector using square brackets []
# The third element in y
> y[3]
[1] 5
# Including negative sign will extract all elements except the # elements
> y
[1] 1 3 5 7 9
> y[-3]
[1] 1 3 7 9
# The first three elements
[1] 1 3 7 9
> y[1:3]
# The first and fifth elements
[1] 1 3 5
> y[c(1, 5)]
[1] 1 9
# The elements without first and fifth elements
> y[-c(1, 5)]
[1] 3 5 7
# The elements that are less than 6
> y[y<6]
[1] 1 3 5

# Create a matrix of values using the "matrix" command
# Set the matrix value from one to nine
# Set "nrow" argument to 3, let R know matrix with three rows and hence three
columns
# Set byrow equal to TRUE, note that this must be capital letters, let R knoow to
enter these elements row-wise
> matrix(c(1,2,3,4,5,6,7,8,9), nrow=3, byrow=TRUE)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
# Set byrow equal to FALSE, note that this must be capital letters, let R knoow
to enter these elements column-wise
> matrix(c(1,2,3,4,5,6,7,8,9), nrow=3, byrow=FALSE)
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> mat <- matrix(c(1,2,3,4,5,6,7,8,9), nrow=3, byrow=TRUE)
> mat
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
# Using the square brackets to extract certain elements in the matrix
> mat[1, 2]
[1] 2
> mat[c(1, 3), 2]
[1] 2 8
# Leaving a row blank or empty will extract that entire row or column
> mat[2,]
[1] 4 5 6
> mat[,1]
[1] 1 4 7
# Preform element-wise addition, subtraction, multiplication or division
> mat*10
     [,1] [,2] [,3]
[1,]   10   20   30
[2,]   40   50   60
[3,]   70   80   90
```

Reference:

Create and Work with Vectors and Matrices in R | R Tutorial 1.4 | MarinStatslectures

# Week 2

## Video 5

### Importing Data from Excel into R (both .csv and .txt files)

```
# Save the data file as comma separated value (.csv) or tab delimited text file
(.txt)
# The ExcelDataCSV.csv file will open in Excel by default
# We can also open this file by any text editor that we like
# Make sure to set the working directory to your CSV file locate in
> setwd("XXXXX/Week2/Video5")
# Access the help menu by "help" command or question mark ? before the command
> help(read.csv)
> ?read.csv

# read.csv command
# First argument to secify is file path to find the file, or we can use
"file.choose()" command to select the file by a popping up menu
# "header" argument setting to TRUE or T let R know that the first row of our
dataset are variable names or headers.
# If first row does not contain variable names, set it to FALSE or F
> data1 <- read.csv(file.choose(), header=T)
# By hitting the "enter" or "return" key on our keyboard and select the file we
wish to import
> data1
   LungCap Age Height Smoke Gender Caesarean
1    6.475   6   62.1    no   male        no
2   10.125  18   74.7   yes female        no
3    9.550  16   69.7    no female       yes
4   11.125  14   71.0    no   male        no
5    4.800   5   56.9    no   male        no
6    6.225  11   58.7    no female        no
7    4.950   8   63.3    no   male       yes
8    7.325  11   70.4    no   male        no
9    8.875  15   70.5    no   male        no
10   6.800  11   59.2    no   male        no
# The worksapce shows that there are 10 obsevations on 6 variables

# Use more generic "read.table" command
# Use the "sep" argumennt in quotations let R know that these are comma separated
values
> data2 <- read.table(file.choose(), header=T, sep=",")
> data2
   LungCap Age Height Smoke Gender Caesarean
1    6.475   6   62.1    no   male        no
2   10.125  18   74.7   yes female        no
3    9.550  16   69.7    no female       yes
4   11.125  14   71.0    no   male        no
5    4.800   5   56.9    no   male        no
6    6.225  11   58.7    no female        no
7    4.950   8   63.3    no   male       yes
8    7.325  11   70.4    no   male        no
9    8.875  15   70.5    no   male        no
10   6.800  11   59.2    no   male        no

# The ExcelDataTAB.txt file is a tab delimited text file
# Use "read.delim" command in R to import a tab-delimited file
> data3 <- read.delim(file.choose(), header=T)
> data3
   LungCap Age Height Smoke Gender Caesarean
1    6.475   6   62.1    no   male        no
2   10.125  18   74.7   yes female        no
3    9.550  16   69.7    no female       yes
4   11.125  14   71.0    no   male        no
5    4.800   5   56.9    no   male        no
6    6.225  11   58.7    no female        no
7    4.950   8   63.3    no   male       yes
```

```
8    7.325  11   70.4    no   male         no
9    8.875  15   70.5    no   male         no
10   6.800  11   59.2    no   male         no
# Use more generic "read.table" command
# Use the "sep" argumennt equal to back-slash t (\t) let R know that these are
tab-delimited file
> data4 <- read.table(file.choose(), header=T, sep="\t")
> data4
   LungCap Age Height Smoke Gender Caesarean
1    6.475   6   62.1    no   male         no
2   10.125  18   74.7   yes female         no
3    9.550  16   69.7    no female        yes
4   11.125  14   71.0    no   male         no
5    4.800   5   56.9    no   male         no
6    6.225  11   58.7    no female         no
7    4.950   8   63.3    no   male        yes
8    7.325  11   70.4    no   male         no
9    8.875  15   70.5    no   male         no
10   6.800  11   59.2    no   male         no
```

---

Reference:

[Import Data, Copy Data from Excel to R CSV & TXT Files | R Tutorial 1.5 | MarinStatsLectures]

---

# Week 2

## Video 6

### How to Import Excel File into R: Using RStudio readxl Built-in Package and Menu

```
# readxl package can import both .xlsx and .xls files
# This packages is pre-installed in RStudio

# Example 1:
# For the first worksheet
# Click File -> Import Dataset -> From Excel... or Click Import Dataset -> From
Excel... in Environment workspace
# Enter the URL if we want to import data from the web or selec browse if we save
the data in our local space
# ******* Make sure the file cannot put under Chinese folder name *******

# Change the "Name" in Import Options to what you want
# By default, R will open up the first worksheet, select the other worksheet if
we wish in "Sheet" options
# "Range" option allows you to select certain rows or columns to import
# "Max Rows" option allows you to limit the number of rows of data that get
imported
# "Skip" option allows you to skip rows on the import, setting this to 1 will
skip the first row, setting this to 2 will skip the first two row and so on
# "NA" option is how we deal with missing values
# Tell R how what our coding is for missing value by filling that in here
# For example, fill in *** in "NA" will change the *** to NA
# Ticking "First Row as Names" box let R knows that our first row of our data set
is the variable name
# Ticking "Open Data Viewer" gives us a data view once the data is imported in
RStudio

# Notice that read Excel doesn't always get the variable types correct
# For example, the disease variable coded using ones and zeros for yes and no
although this variable is actually a factor or categorical. Therefore, click on
the little triangle in "Disease" column and change it to character
# Setting variable to "skip" is going to have our skip it or not import that
variable when importing the data

# The "Code Preview" is the code that we would enter into the console in order to
import the data from the command line.
> library(readxl)
> LungCapData <- read_excel("C:/Users/Jeff/Week2/Video6/Excel Data File.xlsx",
+     sheet = "LungCapData", col_types = c("numeric",
+         "skip", "numeric", "text", "text",
+         "text", "text"), na = "***")
> View(LungCapData)

# Example 2:
# For the second worksheet
# "Range" option allows you to select certain cells to import
> library(readxl)
> OtherData <- read_excel("C:/Users/Jeff/Week2/Video6/Excel Data File.xlsx",
+     sheet = "AnotherDataset", range = "B3:E11")
> View(OtherData)
```

Reference:

Importing/Reading Excel data into R using RStudio (readxl) | R Tutorial 1.5b | MarinStatsLectures

# Week 2

## Video 7

### How to Export Data from R

```
# Import data
> DataToExport = read.table("./DataToExport.csv", header=T, sep=',')
> DataToExport
          Subject Age Gender Score
1 Dave.Andreychuk  53   male  80.5
2     Jon.Stewart  54   male  82.1
3        Jane.Doe  38 female  75.9
4  Amelia.Earhart 119 female  90.0
5    Donald.Trump  70   male -25.5
6   Sidney.Crosby  28   male  87.2
7   Oprah.Winfrey  62 female  88.8
8      Steve.Jobs  61   male  91.1
```

```
# The most flexible command for exporting data is the "write.table" command
# We can use the write.table command to export data, to many formats
> ?write.table

# Save the file in our current working directory, name it "ExportedFileName", and
save as a .CSV file format
> write.table(DataToExport, file="ExportedFileName.csv", sep=",")

# Export without row names by setting the "row.names=FALSE" also note, that this
will over-write the previous file without giving us a warn
> write.table(DataToExport, file="ExportedFileName.csv", row.names=F, sep=",")

# Specify the path for where to save the file instead
> write.table(DataToExport,
+           file="../ExportedFileName.csv",
+           row.names=F, sep=",")

# Write.csv does the same, just dont need to specify sep=","
> write.csv(DataToExport,
+         file="./ExportedFileName.csv",
+         row.names=F)

# Save as tab-delim txt file, setting sep="\t" and file extension to .txt
> write.table(DataToExport,
+           file="./ExportedFileName.txt",
+           row.names=F, sep="\t")

# Save it as space-delimited by setting sep=" "
> write.table(DataToExport,
+           file="./ExportedFileNameSpace",
+           row.names=F, sep=" ")

# "write.csv2" command is used some places in Western Europe and would use a
comma for a deciaml point and a semicolon for a separator rather than a comma
```

---

Reference:

[Export Data from R (csv , txt and other formats) | R Tutorial 1.6 | MarinStatsLectures]

[R Script]

---

# Week 3

## Video 8

### Getting Started with Data in R, Part I

```
# read.table
# Read the data into R and save it as "Data1"
# Specify the "path" to the file in quoatations
# Set "header" argument to TRUE letting R know that the first row of our data is
headers or variables name
# Set "sep" argument letting R know how our observations are separated
# "\t" " " ","
> Data1 <- read.table(file="./LungCapData.txt", header=T, sep="\t")

# Or using the "file.choose" command to select the file
> Data2 <- read.table(file.choose(), header=T, sep="\t")

# Only RStudio Import Dataset
# Select Import Dataset -> From Text File -> Select File -> Drop-down Menu
Select, Heading = Yes, Separator = Tab, Decimal = Period, Quote = Double quote
> LungCapData <- read.delim("./LungCapData.txt")

# Remove command
> rm(Data1)
> rm(Data2)

# dim command, let us know the dimensions of the data in R
> dim(LungCapData)
[1] 725   6

# head command, see the first six rows
> head(LungCapData)
  LungCap Age Height Smoke Gender Caesarean
1   6.475   6   62.1    no   male        no
2  10.125  18   74.7   yes female        no
3   9.550  16   69.7    no female       yes
4  11.125  14   71.0    no   male        no
5   4.800   5   56.9    no   male        no
6   6.225  11   58.7    no female        no

# tail command, see the last six rows
> tail(LungCapData)
    LungCap Age Height Smoke Gender Caesarean
720   7.325   9   66.3    no   male        no
721   5.725   9   56.0    no female        no
722   9.050  18   72.0   yes   male       yes
723   3.850  11   60.5   yes female        no
724   9.825  15   64.9    no female        no
725   7.100  10   67.7    no   male        no

# Review
> LungCapData[c(5,6,7,8,9),]
  LungCap Age Height Smoke Gender Caesarean
5   4.800   5   56.9    no   male        no
6   6.225  11   58.7    no female        no
7   4.950   8   63.3    no   male       yes
8   7.325  11   70.4    no   male        no
9   8.875  15   70.5    no   male        no
> LungCapData[5:9,]
  LungCap Age Height Smoke Gender Caesarean
5   4.800   5   56.9    no   male        no
6   6.225  11   58.7    no female        no
7   4.950   8   63.3    no   male       yes
8   7.325  11   70.4    no   male        no
9   8.875  15   70.5    no   male        no
> LungCapData[-(4:722),]
```

```
    LungCap Age Height Smoke Gender Caesarean
1     6.475   6   62.1    no   male        no
2    10.125  18   74.7   yes female        no
3     9.550  16   69.7    no female       yes
723   3.850  11   60.5   yes female        no
724   9.825  15   64.9    no female        no
725   7.100  10   67.7    no   male        no

# Check the variable names using the "names" command
> names(LungCapData)
[1] "LungCap"   "Age"       "Height"    "Smoke"     "Gender"    "Caesarean"
```

---

Reference:

[Importing , Checking and Working with Data in R | R Tutorial 1.7 | MarinStatsLectures]()

---

# Week 3

## Video 9

### Working with Data in R, Part II

```
# Import the LungCapData.txt as LungCapData first
> LungCapData <- read.delim("./LungCapData.txt", stringsAsFactors=TRUE)

# Using $ to extract variables from the object
> LungCapData$Age

# "mean" function
> mean(LungCapData$Age)
[1] 12.3269

# Attach data
# Pro: Attaching data by their name without $
# Con: Be overwritten more easily
> attach(LungCapData)
> mean(Age)
[1] 12.3269

# "detach" commmand
> detach(LungCapData)
> Age
Error: object 'Age' not found

# Checking the type of variable using the "class" command
> attach(LungCapData)
> names(LungCapData)
[1] "LungCap"   "Age"       "Height"    "Smoke"     "Gender"    "Caesarean"
> class(LungCap)
[1] "numeric"
> class(Age)
[1] "integer"
```

```
> class(Height)
[1] "numeric"
> class(Smoke)
[1] "factor"
> class(Gender)
[1] "factor"
> class(Caesarean)
[1] "factor"
# Note that if the Smoke, Gender and Caesarean shows character, it is caused by
read.table command

# "levels" command to ask what the different levels or categories are for this
factor
> levels(Smoke)
[1] "no"  "yes"
> levels(Gender)
[1] "female" "male"
> levels(Gender)
[1] "female" "male"

# "summary" command
> summary(LungCapData)
    LungCap            Age            Height       Smoke        Gender
 Caesarean
 Min.   : 0.507   Min.   : 3.00   Min.   :45.30   no :648   female:358   no :561

 1st Qu.: 6.150   1st Qu.: 9.00   1st Qu.:59.90   yes: 77   male  :367   yes:164

 Median : 8.000   Median :13.00   Median :65.40

 Mean   : 7.863   Mean   :12.33   Mean   :64.84

 3rd Qu.: 9.800   3rd Qu.:15.00   3rd Qu.:70.30

 Max.   :14.675   Max.   :19.00   Max.   :81.80

# Convert x to a cateogrical variable or a factor using "as.factor" command
> x <- c(0,1,1,1,0,0,0,0,0,0)
> class(x)
[1] "numeric"
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    0.00    0.00    0.30    0.75    1.00
> x <- as.factor(x)
> class(x)
[1] "factor"
# Report frequency for factor
> summary(x)
0 1
7 3
```

Reference:

[Working with Variables and Data in R | R Tutorial 1.8 | MarinStatslectures]

# Week 3

## Video 10

### Subsetting Data Using Square Brackets in R

```
# Import the LungCapData.txt as LungCapData first
> LungCapData <- read.delim("./LungCapData.txt", stringsAsFactors=TRUE)

# Review
> dim(LungCapData)
[1] 725   6
> length(Age)
[1] 725
> Age[11:14]
[1] 19 17 12 10
> LungCapData[11:14, ]
   LungCap Age Height Smoke Gender Caesarean
11  11.500  19   76.4    no   male       yes
12  10.925  17   71.7    no   male        no
13   6.525  12   57.5    no   male        no
14   6.000  10   61.1    no female        no


# Calculate the mean age but only for females
> mean(Age[Gender=="female"])
[1] 12.44972

# = (equal sign) is to assign values to objects
# == (double equal sign) is used to represent the equality in a mathemaical
sense

# Store all the female data as FemData, and male data as MaleData
> FemData <- LungCapData[Gender=="female", ]
> MaleData <- LungCapData[Gender=="male", ]

# Check
> dim(FemData)
[1] 358   6
> dim(MaleData)
[1] 367   6
> summary(Gender)
female   male
   358    367

# Pull out data for male who are over 15 years old
> MaleOver15 <- LungCapData[Gender=="male" & Age>15, ]
```

Reference:

[Subsetting (Sort/Select) Data in R with Square Brackets | R Tutorial 1.9| MarinStatsLectures]

# Week 4

## Video 11

### Logic Statements, and a Few Other Random But Useful Commands in R

```
# Import the LungCapData.txt as LungCapData first
> LungCapData <- read.delim("./LungCapData.txt", stringsAsFactors=TRUE)
> LungCapData[1:5, ]
  LungCap Age Height Smoke Gender Caesarean
1   6.475   6   62.1    no   male        no
2  10.125  18   74.7   yes female        no
3   9.550  16   69.7    no female       yes
4  11.125  14   71.0    no   male        no
5   4.800   5   56.9    no   male        no

# "logic" command
> Age[1:5]
[1]  6 18 16 14  5
> temp <- Age[1:5] > 15
> temp
[1] FALSE  TRUE  TRUE FALSE FALSE

# "as.numeric()" command
> temp2 <- as.numeric(Age[1:5]>15)
> temp2
[1] 0 1 1 0 0

# multiple logical statements
> FemSmoke <- Gender=="female" & Smoke=="yes"
> FemSmoke[1:5]
[1] FALSE  TRUE FALSE FALSE FALSE

# Attach vectors or matrices in a colummn-wise using "cbind" command as well as
row-wise using "rbind" command
> MoreData <- cbind(LungCapData, FemSmoke)
> MoreData[1:5, ]
  LungCap Age Height Smoke Gender Caesarean FemSmoke
1   6.475   6   62.1    no   male        no    FALSE
2  10.125  18   74.7   yes female        no     TRUE
3   9.550  16   69.7    no female       yes    FALSE
4  11.125  14   71.0    no   male        no    FALSE
5   4.800   5   56.9    no   male        no    FALSE

# Remove all objects in command
> rm(list=ls())
```

Reference:

Logic Statements (TRUE/FALSE), cbind and rbind Functions in R | R Tutorial 1.10| MarinStatsLectures

# Week 4

## Video 12

### Setting Up Your Working Directory in R

```
# Import the LungCapData.txt as LungCapData first
> LungCapData <- read.delim("./LungCapData.txt", stringsAsFactors=TRUE)
> attach(LungCapData)

# "getwd" command
> getwd()

# "setwd" command
> setwd("../Video12")
# or save path in a variable
> projectWD <- "../Video12"
> setwd(projectWD)

# Session -> Set Working Directory -> Choose Directory
> MeanAge <- mean(Age)
> x <- c(1,2,3,4,5)
> y <- 14
> z = summary(LungCapData)

# "save.imgae" command save the current workspace image
# .Rdata let R know that this is a workspace image file
# Without specify the path, it will be saved in the current working directory
> save.image("FirstProject.Rdata")

# Another way to save the workspace image
# Session -> Save Workspace As...

# Same as rm(list=ls())
# Session -> Clear Workspace...

# Quit RStudio
> q()

# Reopen RStudio it is clear
> ls()
character(0)
> setwd("../Week4/Video12/")

# "load" command load previous workspace image
> load("FirstProject.Rdata")
> load(file.choose())
# Session -> Load Workspace...
```

Reference:

Setting Up Working Directory in R | R Tutorial 1.11 | MarinStatsLectures

# Week 4

## Video 13

### Writing Scripts of Code in R

```
# number sign (#) can be used to prompt R to ignore anything that follows

# Create a new script: File -> New File -> R Script
# Or open an existing file in folder File -> Open File


# Submit a code rather than copy a code, place the cursor on the line of code we
would like to have submitted and then click on the "Run" option
# Or a keyboard shortcut "command + Enter" in Mac or "Ctrl + Enter"
# Highligth the code we would like to enter and select "Run"

# comment and uncomment lines: Select Code -> Comment/Uncomment Lines

# Tab key will return to you a list of suggestions or variables of what you may
be looking for

# Save the file: File -> Save As...
```

---

Reference:

[Writing Scripts in R | R Tutorial 1.12 | MarinStatsLectures]

---

# Week 4

## Video 14

### Installing Packages in R

```
# "install.packages" command or using the menus within R to install packages
> help("install.packages")

# Example of installing "epiR" package
> install.packages("epiR")

# Leave it blank and R will return a menu of all packages available
> install.packages()

# load the libary of "epiR" package
> library(epiR)
載入需要的套件：survival
Package epiR 2.0.38 is loaded
Type help(epi.about) for summary information
Type browseVignettes(package = 'epiR') to learn how to use epiR for applied
epidemiological analyses
```

```
# Need to load the library each R session that we would like to be able to use
the package

# Click on CRAN on R website to download R or packages for R
# Select the mirror and enter it
# In leftside, Software -> Packages -> Table of availabe packages...

# help for a specific package
> help(package=epiR)

# Remove a package
> remove.packages("epiR")

# Or using GUI to install and uninstall packages
# Tools -> Install Packages
# Also Check for Package Updates...
# Specify the packages we want to install
```

---

Reference:

[How to Install Packages in R | R Tutorial 1.13 | MarinStatsLectures]()

[Complete list of R packages]()

---